Notas de Aula
Prof. Adriano Milanez

Programação Web I
Responsive Web Development
Front-End Design Engineering
Programação Web Responsiva

[aula 04]
[Atualizado em: 25/08/2025]

[GIT e GITHUB]

01

GIT E GITHUB

GIT E GITHUB

GIT e **GITHUB** são duas tecnologias que todo desenvolvedor deve aprender, independentemente de sua área.

Se você é um desenvolvedor iniciante, pode pensar que esses dois termos significam a mesma coisa, mas são diferentes.

O QUE É O GIT?

O **GIT** é um sistema que ajuda a controlar as mudanças feitas em arquivos ao longo do tempo, como se fosse um “histórico de versões”. Ele é usado principalmente para código, mas pode ser usado com qualquer tipo de arquivo.

💡 **Exemplo simples**

Imagine que você está escrevendo um livro no Microsoft Word, o **GIT** é como um assistente que tira “fotos” de cada versão do seu livro, para que você possa voltar para uma versão anterior se algo der errado.

O QUE É O GIT?

GIT é um sistema de versionamento, criado por *Linus Torvalds*, autor do **Linux**.

É capaz de guardar o histórico de alterações de todos os arquivos dentro de uma pasta, que chamamos de repositório.

Funciona como o “Track changes” do Microsoft Word, mas muito melhor.

Torna-se importante à medida que seu trabalho é colaborativo.

GIT é um software que você instala no computador.

O QUE É O GITHUB?

O **GITHUB** é como uma “rede social” para projetos que usam o **GIT**. Ele permite que você salve seus projetos na internet (em vez de apenas no seu computador), colabore com outras pessoas e mostre seu trabalho ao mundo.

💡 **Exemplo simples**

Pense no **GITHUB** como o “Google Drive” para os projetos que estão sendo controlados pelo **GIT**. Você pode guardar, compartilhar e colaborar com outras pessoas nos seus arquivos.

O QUE É O GITHUB?

GITHUB é um *site* onde você coloca e compartilha repositórios **GIT**.

Utilizado por milhões de pessoas em projetos de código aberto ou fechado.

Útil para colaborar com outros programadores em projetos de ciência de dados.

Existem alternativas, como GitLab e BitBucket.

GITHUB é um site que você acessa na *internet*.

PRINCIPAIS DIFERENÇAS

GIT	GITHUB
É uma ferramenta de controle de versões.	É uma plataforma *online* para hospedar projetos GIT.
Funciona no seu computador, mesmo sem internet.	Precisa de *internet* para acessar.
Gerencia versões e mudanças nos arquivos.	Ajuda a compartilhar e colaborar em projetos.
Não tem interface gráfica, é usado pelo terminal.	Tem uma interface 'amigável' e funcional.

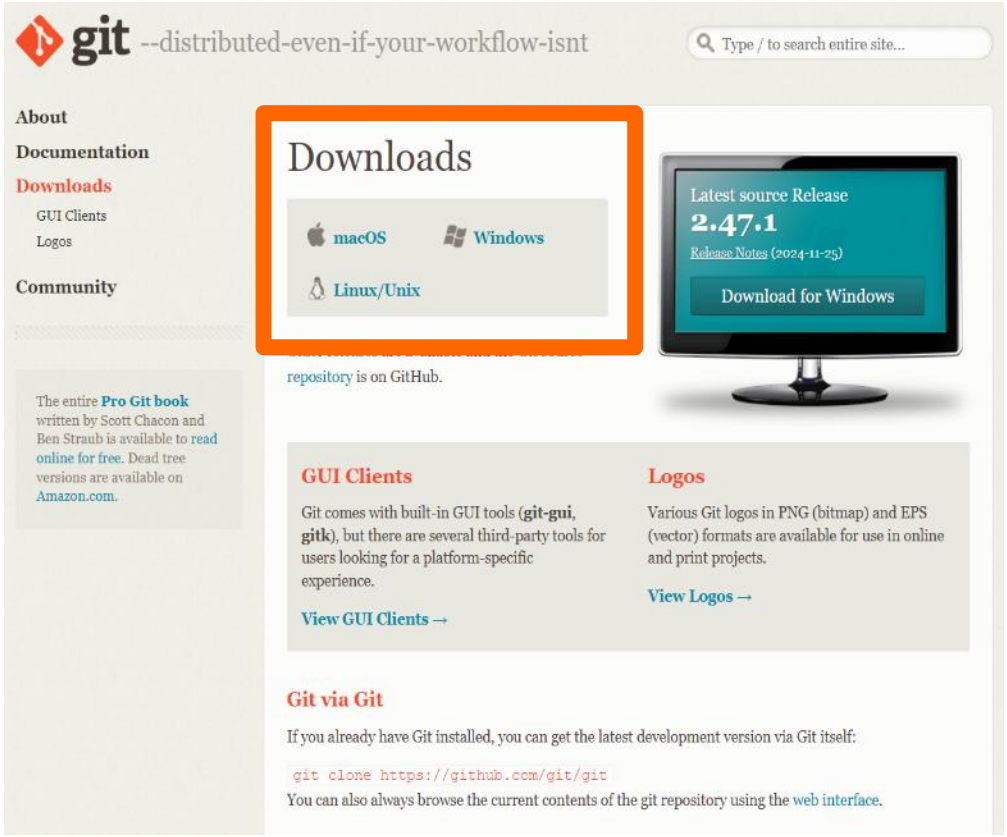
GIT – INSTALAÇÃO

O **GIT** fornece ferramentas de linha de comando para realizar as ações mais comuns em um repositório, como salvar alterações (commits), visualizar histórico de versões e gerenciar *branches*. Ele também é compatível com interfaces gráficas (GUI) de **terceiros**, que tornam essas tarefas mais intuitivas, enquanto a linha de comando continua sendo indispensável para cenários mais avançados.

GIT – INSTALAÇÃO

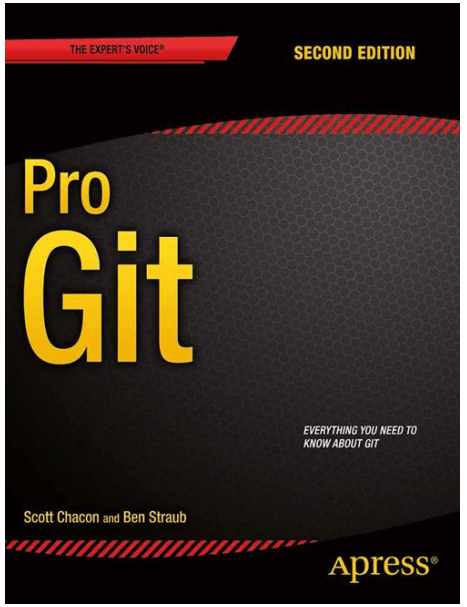
Download para TODAS as plataformas neste *link*

https://git-scm.com/downloads



GIT - TUTORIAL

Existe um *ebook,* em inglês, distribuído pelo próprio GIT, disponível para *download* neste *link* - https://github.com/progit/progit2/releases/download/2.1.439/progit.pdf



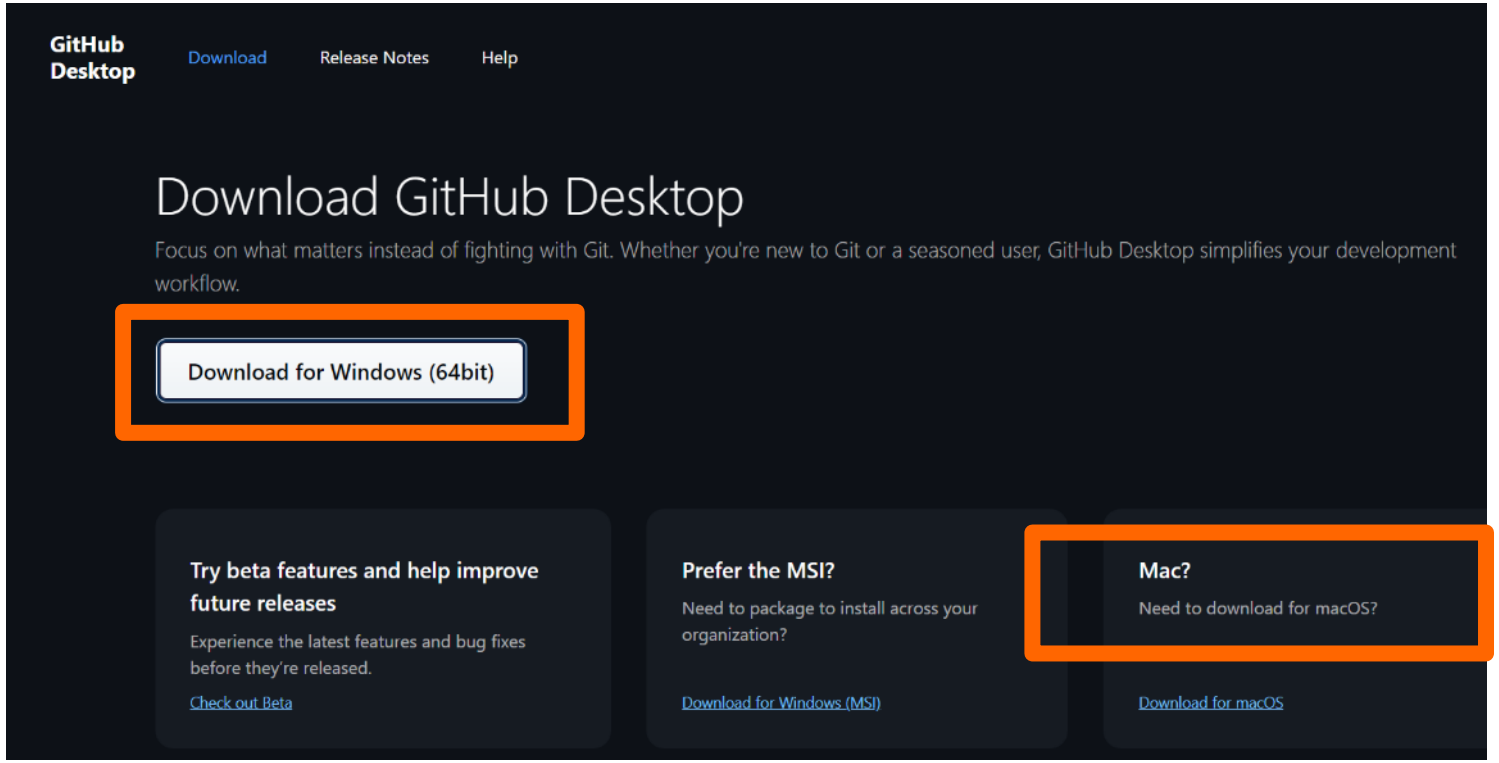
Há também uma versão uma versão em português, porém, não traduzida completamente, disponível em: https://git-scm.com/book/pt-br/v2

GITHUB – INSTALAÇÃO

GITHUB fornece clientes *desktop* que incluem uma interface gráfica para as ações mais comuns em um repositório e atualiza automaticamente para a linha de comando do **GIT** para cenários avançados.

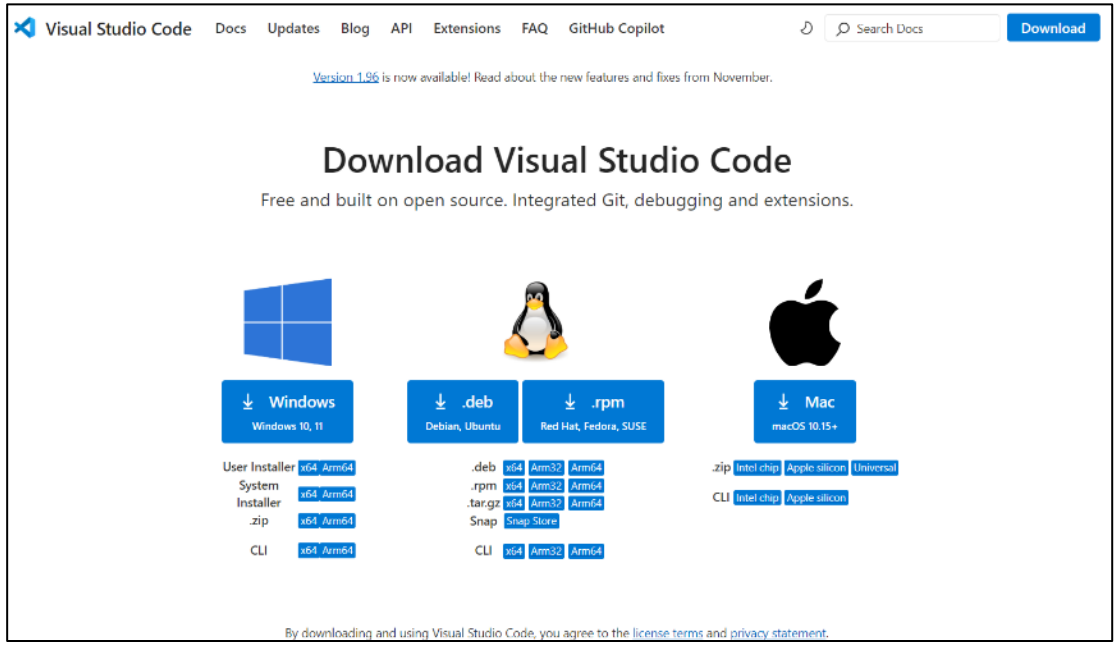
GITHUB – INSTALAÇÃO

Download neste *link* https://desktop.github.com/download/



VSCODE – INSTALAÇÃO

Para nosso tutorial, não vamos fazer *download* e/ou instalação do **GITHUB**, vamos trabalhar com instruções no *prompt* de comando e também utilizando o VSCode, caso não o tenha instalado, pode fazer o *download* e instalação, neste *link* - https://code.visualstudio.com/download.



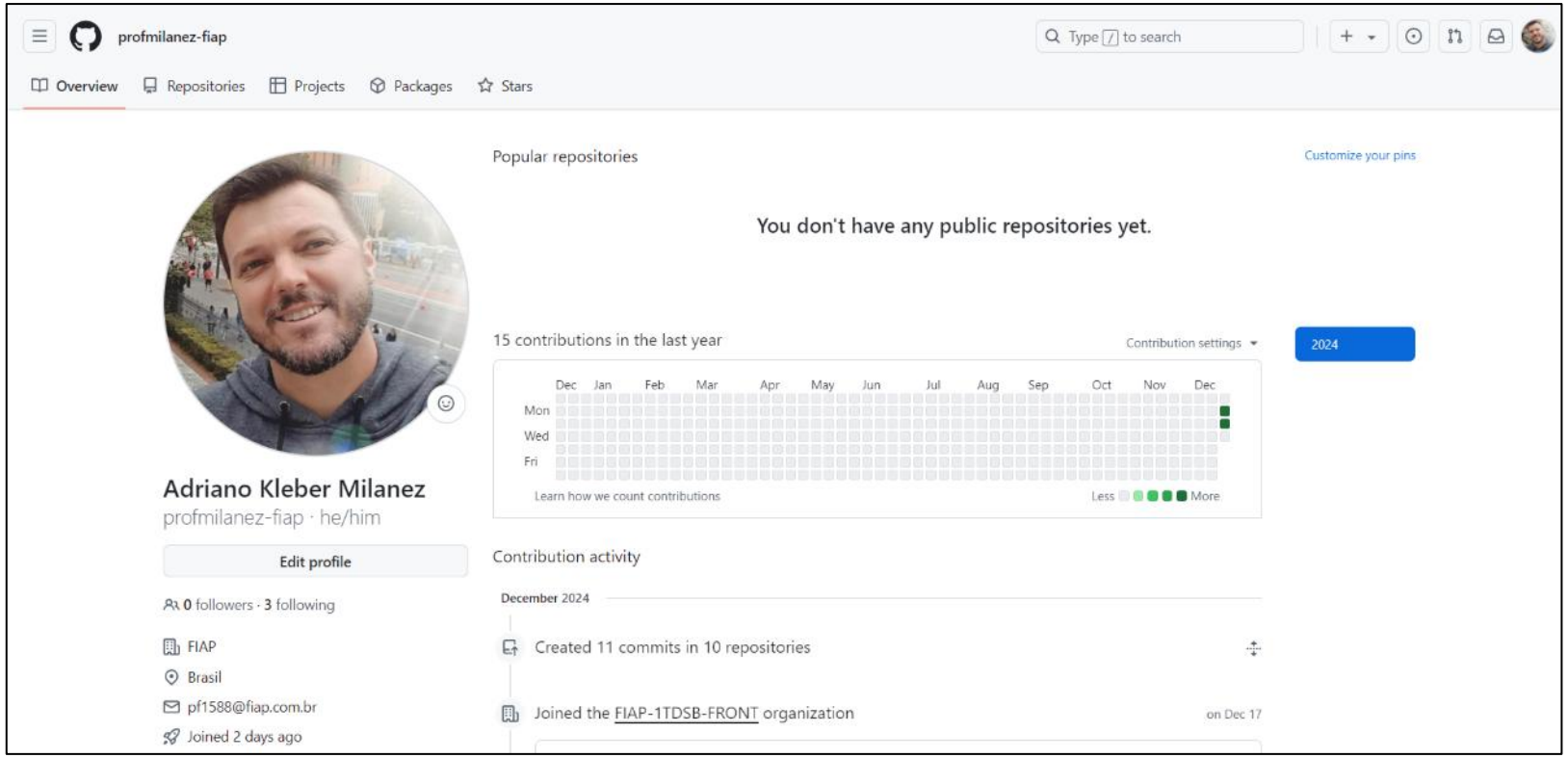
GITHUB – CONTA

Precisamos ter, apenas, uma conta criada no **GITHUB**

- **Não estudante**
  - **Conta existente** https://github.com/login
  - **Criar conta** https://github.com/signup
- **Estudante**
  - **Conta existente** https://education.github.com/discount_requests/application?type=student
  - **Criar conta** https://github.com/signup

GITHUB – PERFIL

Criado a conta e efetuado o login, temos acesso ao perfil.

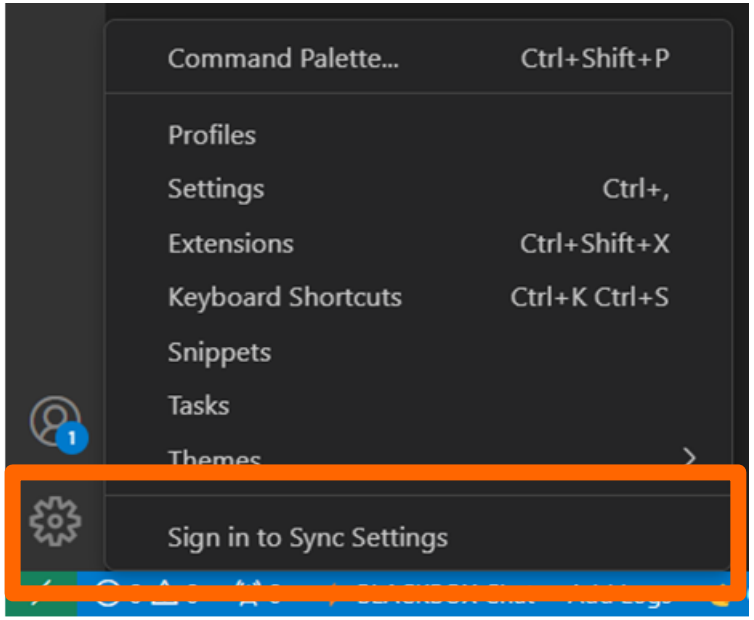


VSCODE E GITHUB

CONFIGURANDO O AMBIENTE

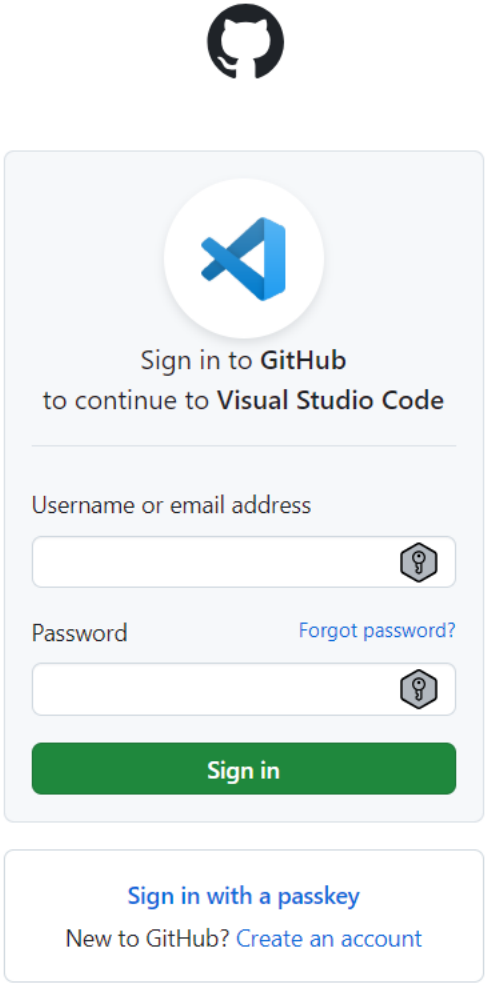
VSCODE E GITHUB – CONFIGURANDO O AMBIENTE

Para usar o **GIT** e o **GITHUB** no **Visual Studio Code** (**VSCode**), certifique-se de que o **GIT** está instalado no seu computador.



Você pode fazer login no **VSCode** com sua conta do GitHub. Para isso, clique no menu de Contas, localizado no canto inferior direito da barra de Atividade.

VSCODE E GITHUB – CONFIGURANDO O AMBIENTE



Uma nova janela deve ser aberta, onde você deve digitar seu usuário/email e sua senha para conectar o **VSCode** com o **GITHUB**.

VSCODE E GITHUB – CONFIGURANDO O AMBIENTE

Com o login correto, você deve receber essa mensagem.

Pronto está configurado e podemos utilizar o **VSCode** para criar nossos códigos e enviar nossos projetos diretamente aos repositórios[1] do **GITHUB**.

1 Um repositório é o elemento mais básico do GitHub. É um lugar onde você pode armazenar seu código, seus arquivos e o histórico de revisão de cada arquivo. Os repositórios podem ter vários colaboradores e podem ser públicos ou particulares, explicaremos mais adiante a diferença entre as visibilidades.

VSCODE E GITHUB – CONFIGURANDO O AMBIENTE

Configurado o ambiente, vamos criar nosso repositório local e seus arquivos para depois fazermos o *upload* ao repositório remoto, podemos fazer isso de duas formas:

1) Através do Terminal (prompt)
2) Através do **VSCode**

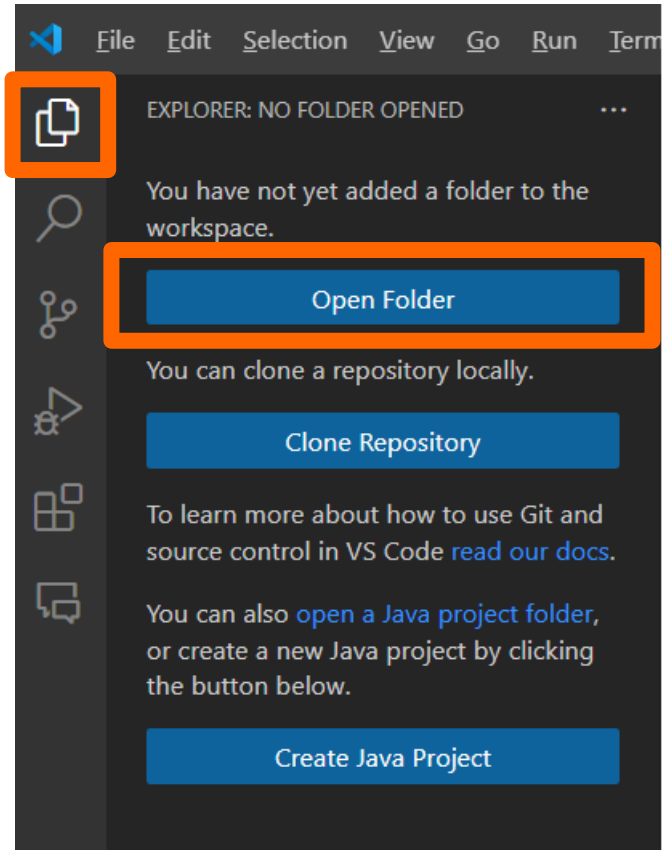
Vamos mostrar as duas formas, percebam que a criação de repositórios, arquivos é comum para ambas as formas, o que vai diferenciar uma da outra é a forma como os arquivos serão enviados ao repositório remoto.

GIT E GITHUB

CRIANDO, COMMITANDO E PUBLICANDO REPOSITÓRIOS

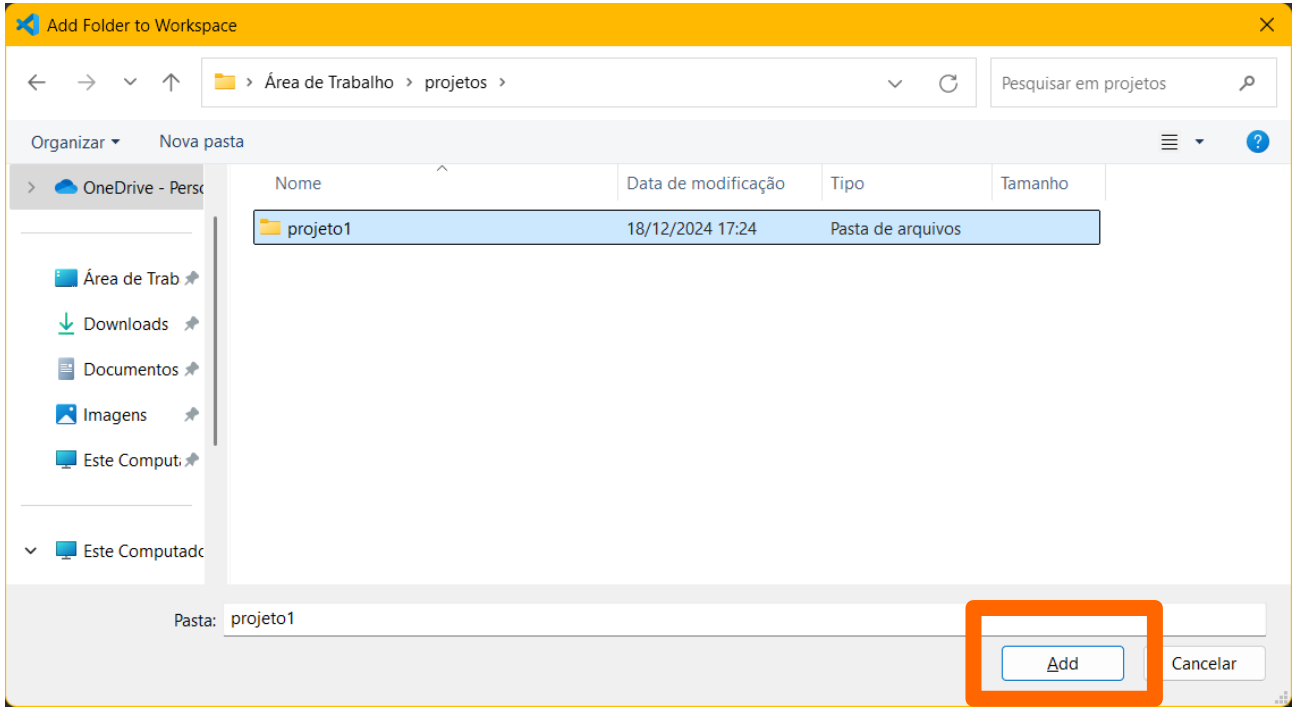
VSCODE – CRIANDO REPOSITÓRIOS LOCAIS

No **VSCode**, vamos criar nossos repositórios (pastas, diretórios) locais.



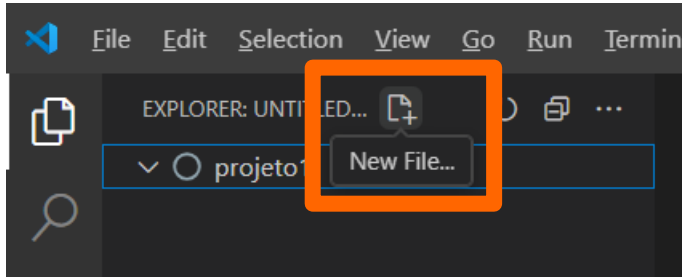
VSCODE – CRIANDO REPOSITÓRIOS LOCAIS

Vamos criar um novo repositório chamado “Projeto1” e na sequência clicamos em “Add”



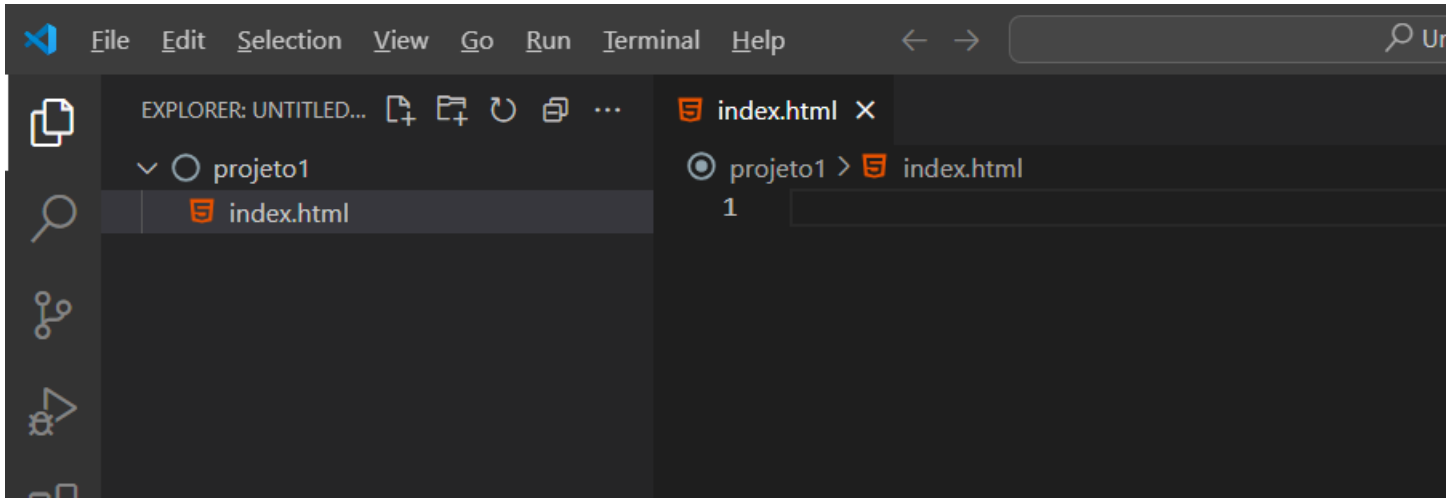
VSCODE – CRIANDO REPOSITÓRIOS LOCAIS

Vamos criar um arquivo, chamado “index.html”, clicando ícone “New File”, como na imagem abaixo



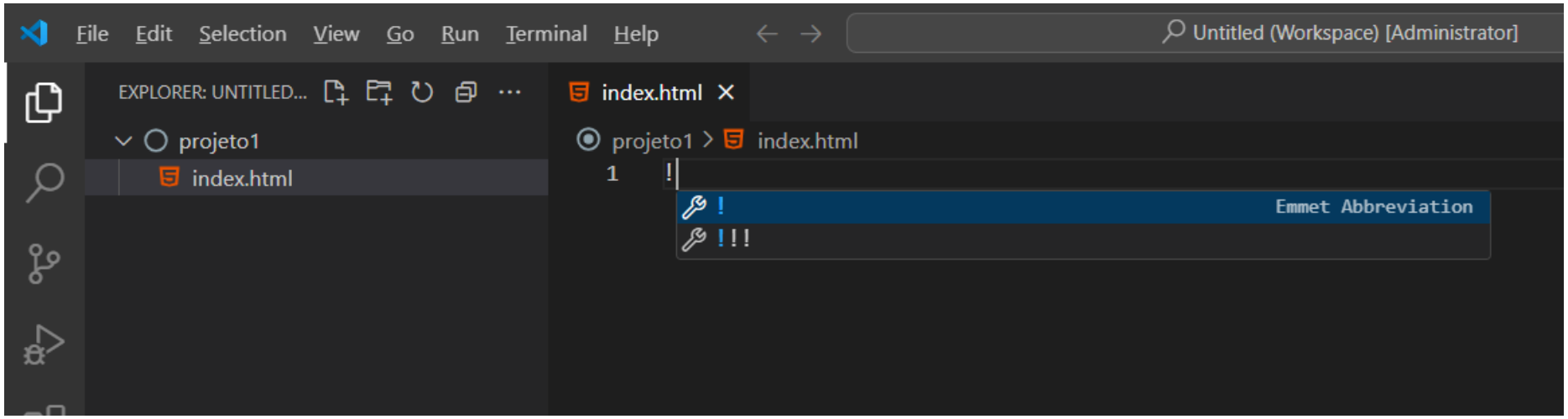
VSCODE – CRIANDO REPOSITÓRIOS LOCAIS

Como resultado temos:



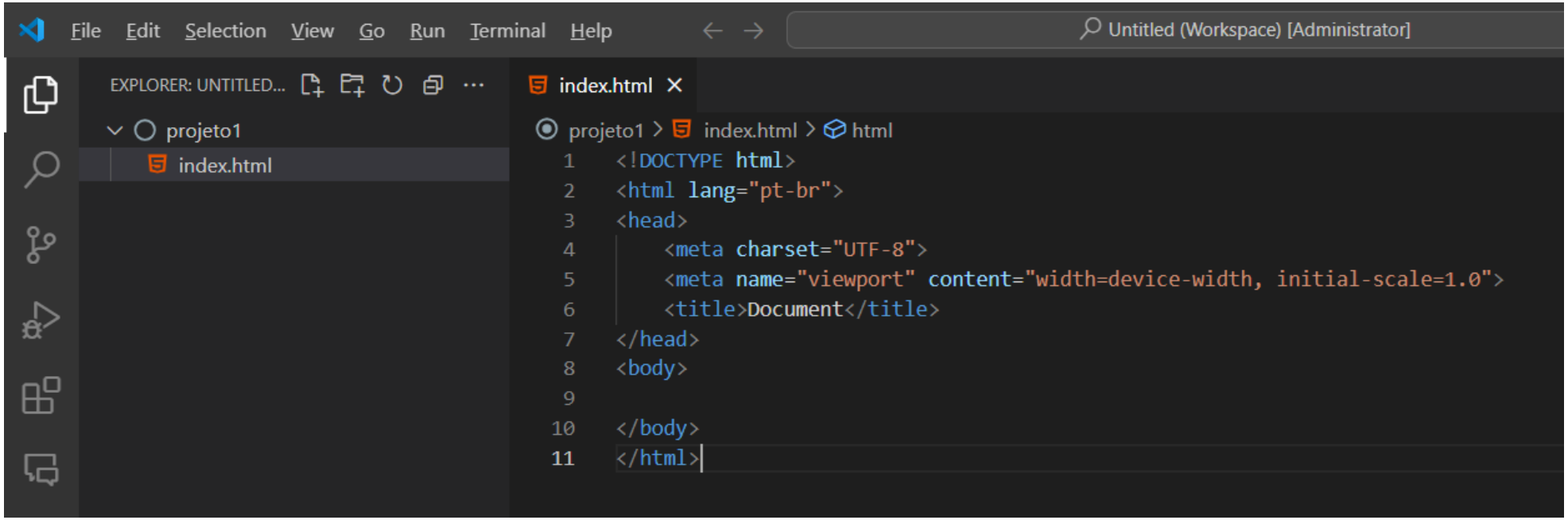
VSCODE – CRIANDO REPOSITÓRIOS LOCAIS

Vamos criar conteúdo digitando uma exclamação e dar ‘enter’



VSCODE – CRIANDO REPOSITÓRIOS LOCAIS

O resultado será um HTML básico.

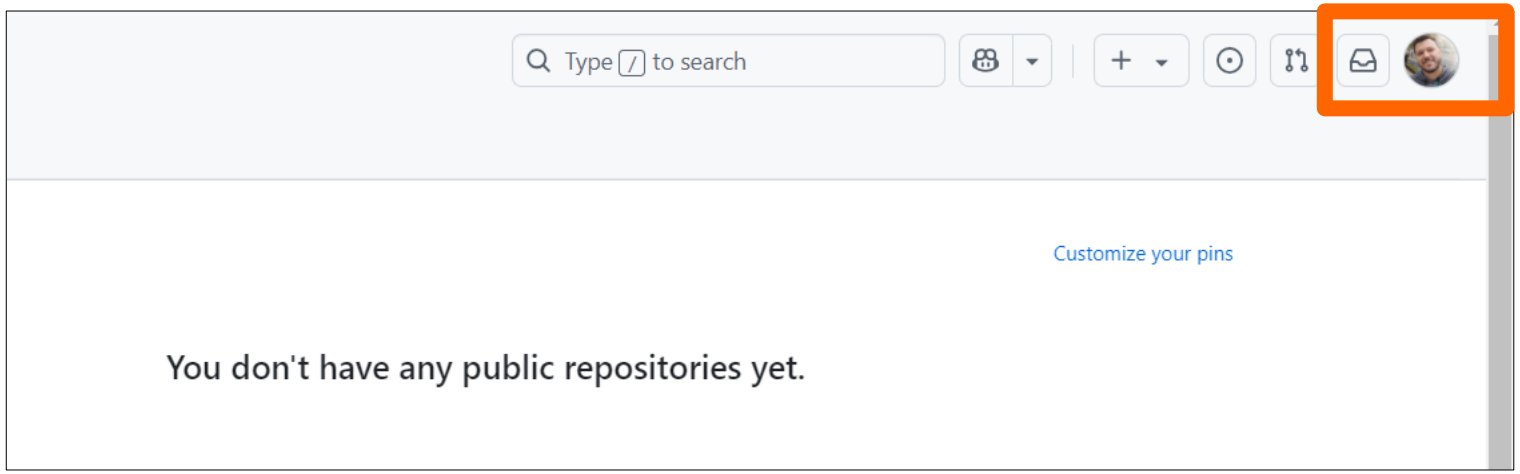


Lembrando que fizemos com um arquivo HTML, mas podemos utilizar QUALQUER tipo de arquivo, *.txt, *.docx, *.xlsx, *.c, *.css, etc.

GITHUB – CRIANDO REPOSITÓRIOS REMOTOS (TERMINAL)

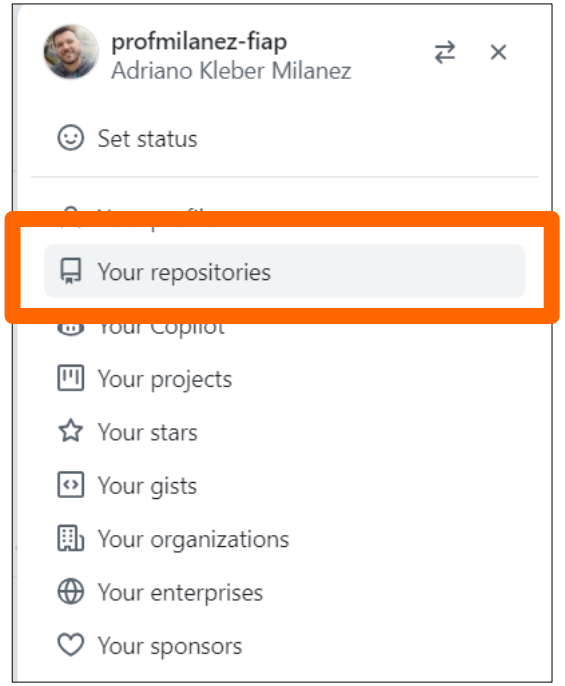
Para utilizarmos o terminal, precisamos, primeiramente ter criado o repositório remoto.

Vamos acessar o perfil no GITHUB, clicamos na foto do perfil, no canto superior direito.



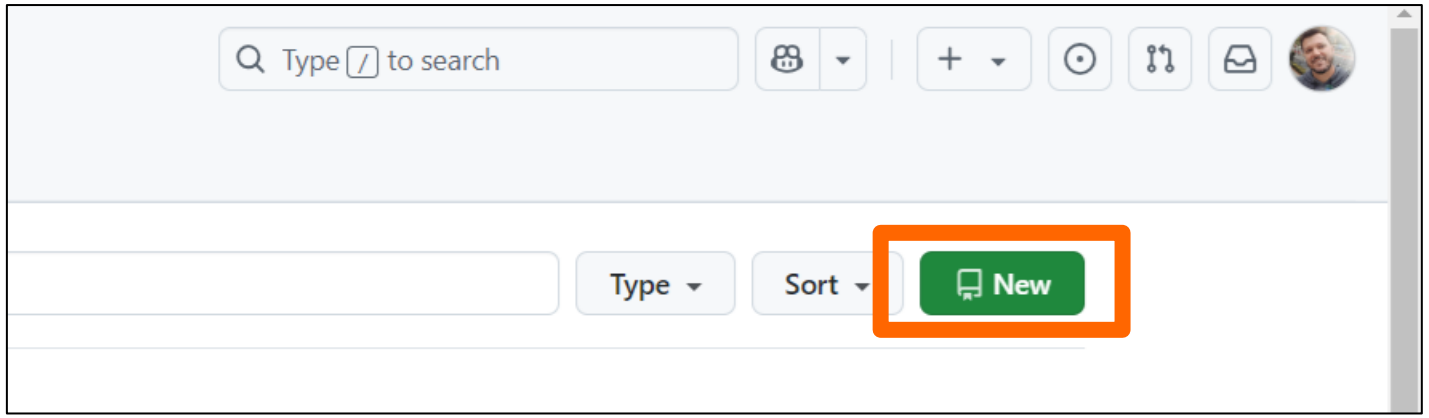
GITHUB – CRIANDO REPOSITÓRIOS REMOTOS (TERMINAL)

Na sequência, clicamos em ‘Your repositories’.

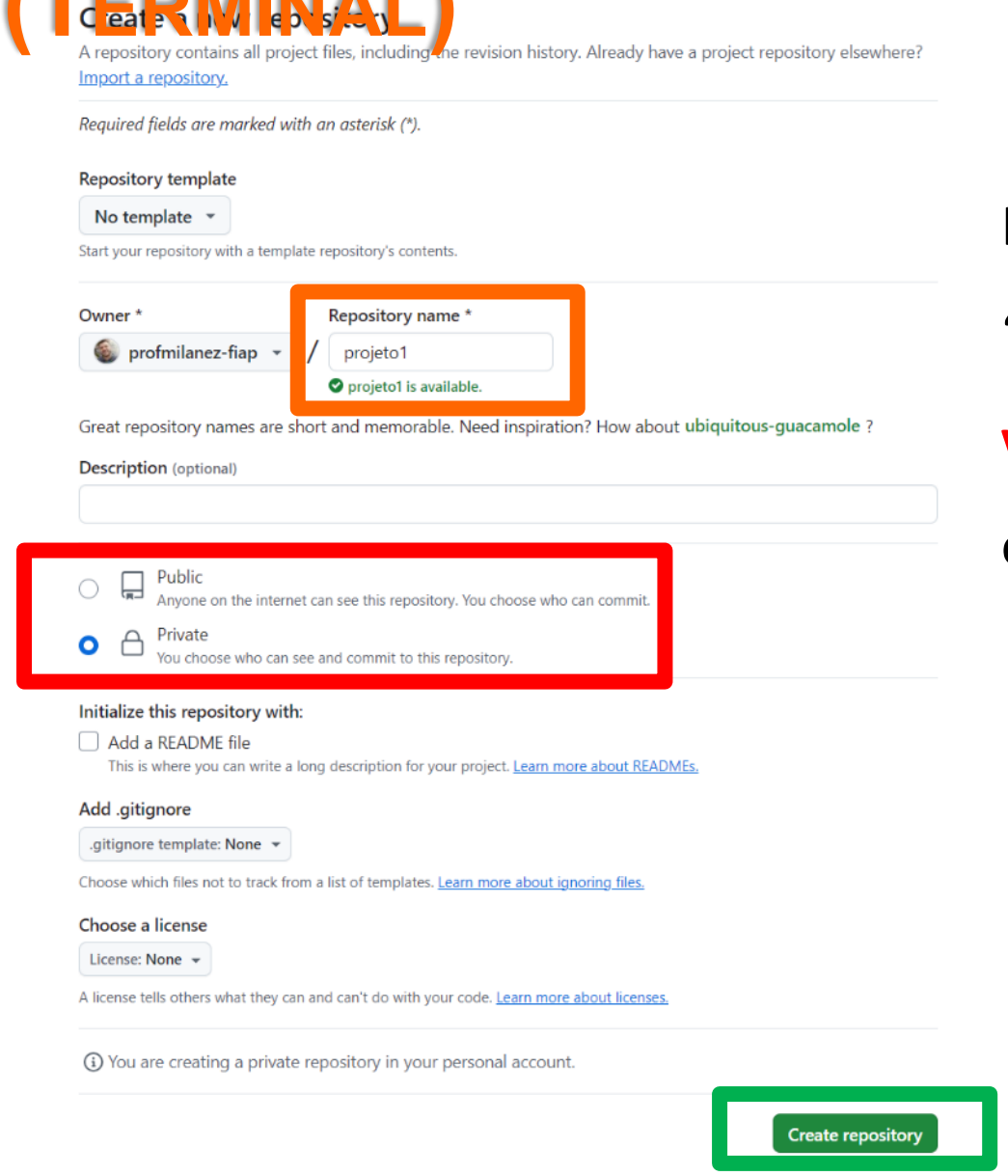


GITHUB – CRIANDO REPOSITÓRIOS REMOTOS (TERMINAL)

E depois clicamos em ‘New’.



GITHUB – CRIANDO REPOSITÓRIOS REMOTOS (TERMINAL)

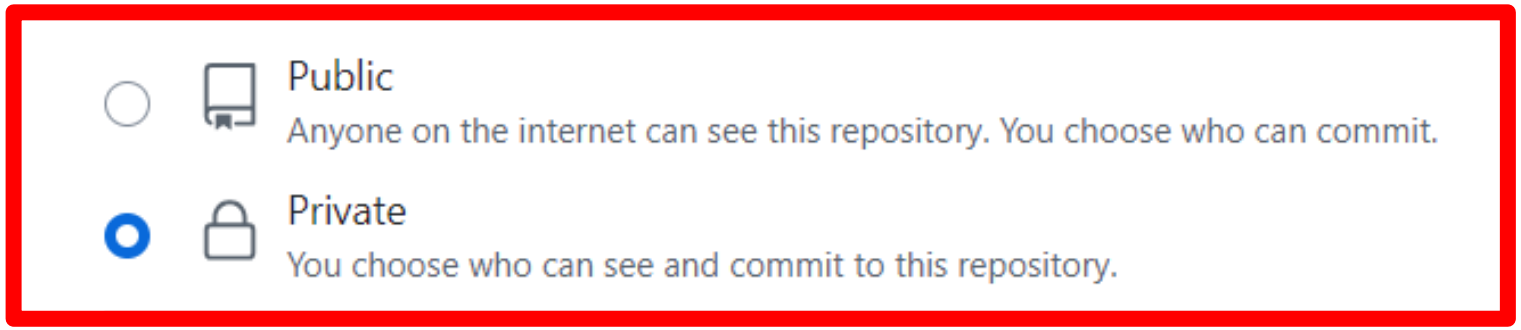


Inserimos o nome do novo repositório em '**Repository name**', e definimos a **visibilidade do repositório** que será criado e clicamos em '**Create repository**'.

GITHUB – CRIANDO REPOSITÓRIOS LOCAIS

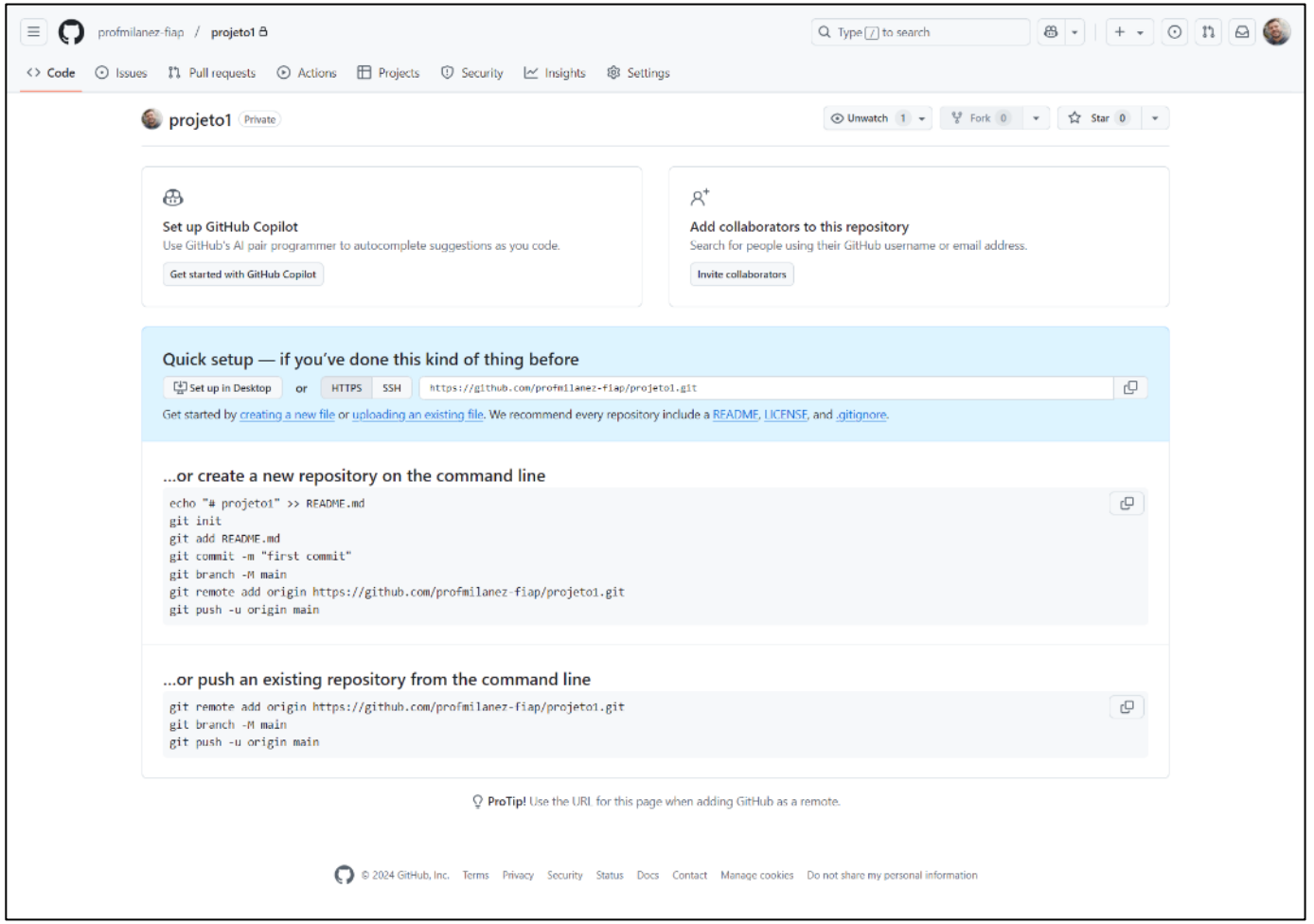
Sobre a visibilidade:

- **Públicos**: Os repositórios públicos são acessíveis a todos na Internet.
- **Privados**: Os repositórios só podem ser acessados por você, pelas pessoas com as quais você compartilha explicitamente o acesso e, para repositórios da organização, por determinados integrantes da organização.



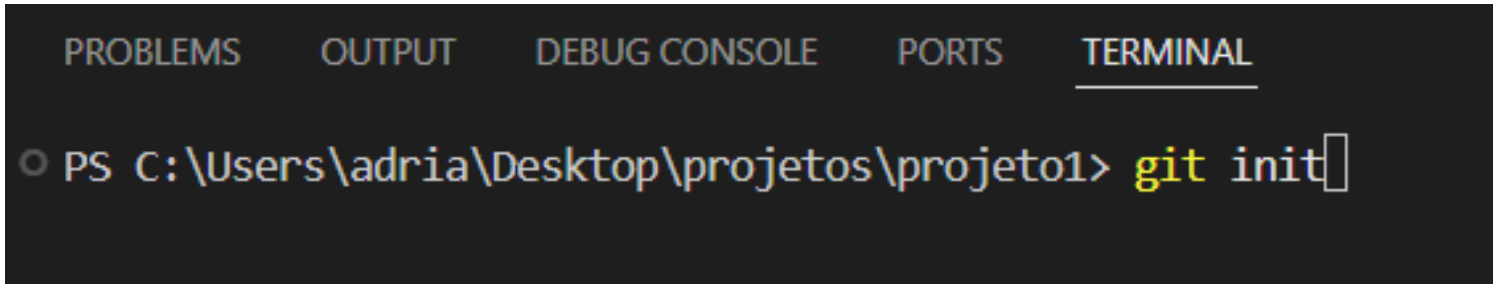
GITHUB – CRIANDO REPOSITÓRIOS REMOTOS (TERMINAL)

Criado o repositório, teremos a seguinte resposta:



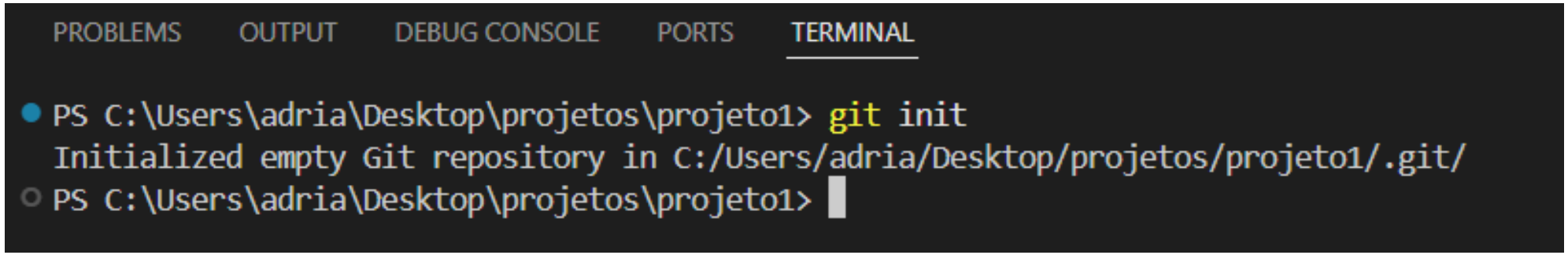
GITHUB – INICIANDO REPOSITÓRIOS LOCAIS (TERMINAL)

Precisamos inicializar o repositório localmente. Vamos abrir o **Terminal**, na pasta que criamos para nosso projeto. Podemos pressionar `CRTL + '` ou `CRTL + SHIFT + '` e digitamos: `git init` e na sequência pressionamos '`enter`'.



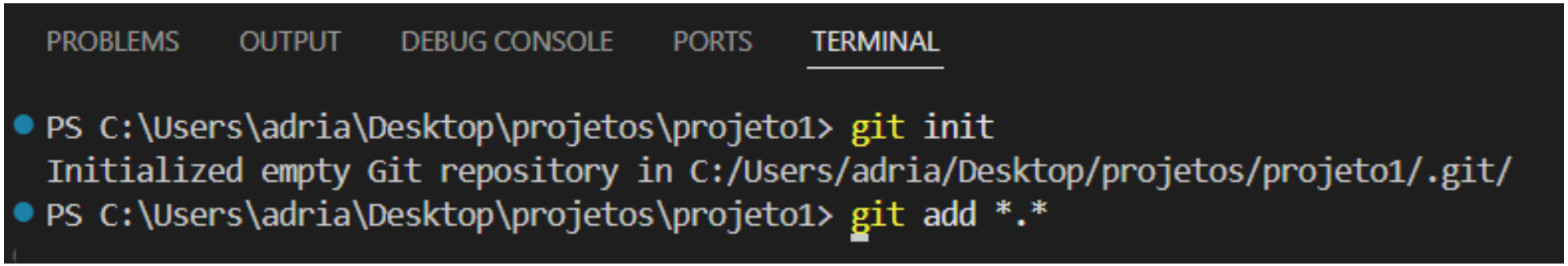
GITHUB – INICIANDO REPOSITÓRIOS LOCAIS (TERMINAL)

Como resposta, temos:



GITHUB – ADICIONANDO ARQUIVOS EM REPOSITÓRIOS LOCAIS (TERMINAL)

Precisamos adicionar o(s) arquivo(s) do nosso projeto, para isso vamos digitar: `git add *.*`, ou `git add *`, isso deverá adicionar todos os arquivos que existem na pasta 'Projeto1' e na sequência pressionamos '`enter`'.



GITHUB – ADICIONANDO ARQUIVOS EM REPOSITÓRIOS LOCAIS (TERMINAL)

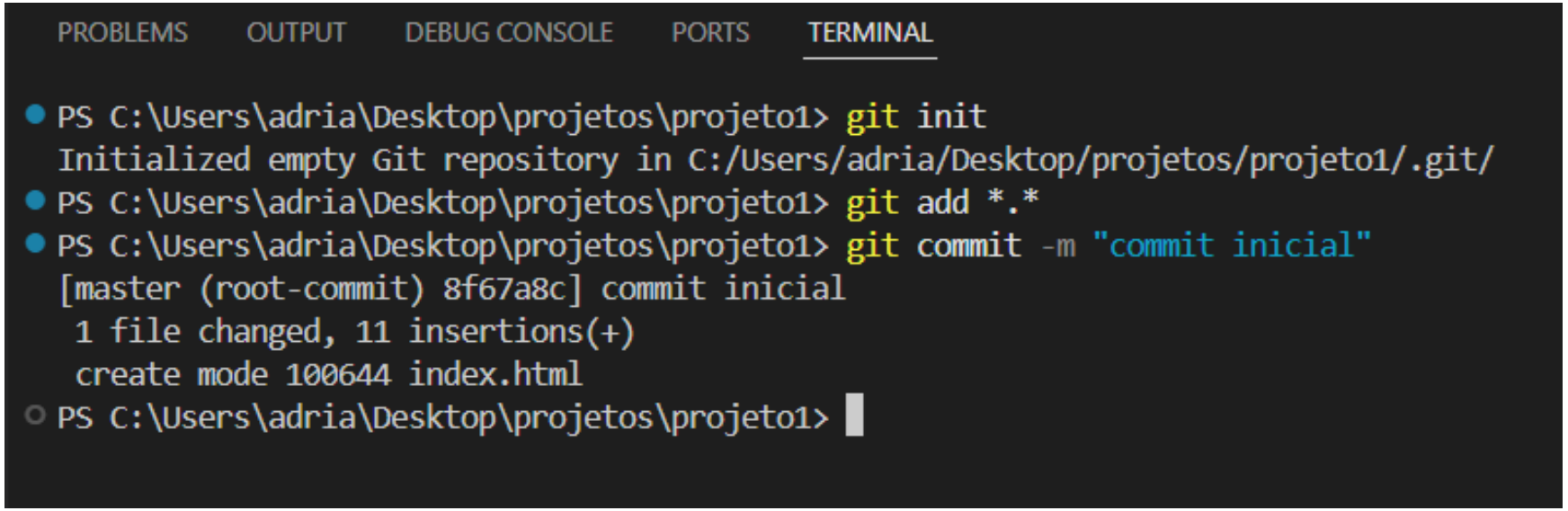
Como resposta, temos:

GITHUB – *COMMITANDO* REPOSITÓRIOS LOCAIS (TERMINAL)

Agora vamos criar nosso primeiro *commit*, para isso vamos digitar: `git commit -m "commit inicial"` e na sequência pressionamos ‘`enter`’.

GITHUB – COMMITANDO REPOSITÓRIOS LOCAIS (TERMINAL)

Como resposta, temos:



GITHUB – *COMMITANDO* REPOSITÓRIOS LOCAIS (TERMINAL)

É importante criar uma mensagem de *commit* para:

- Facilitar a comunicação entre os colaboradores,
- Tornar o código mais rastreável e compreensível,
- Ajudar a encontrar *commits* relevantes,
- Permitir uma colaboração mais eficiente,
- Servir como documentação automática do código.

Uma mensagem de *commit* é uma parte importante da documentação do **GIT**, pois registra as alterações feitas no código ao longo do tempo.

GITHUB – *COMMITANDO* REPOSITÓRIOS LOCAIS (TERMINAL)

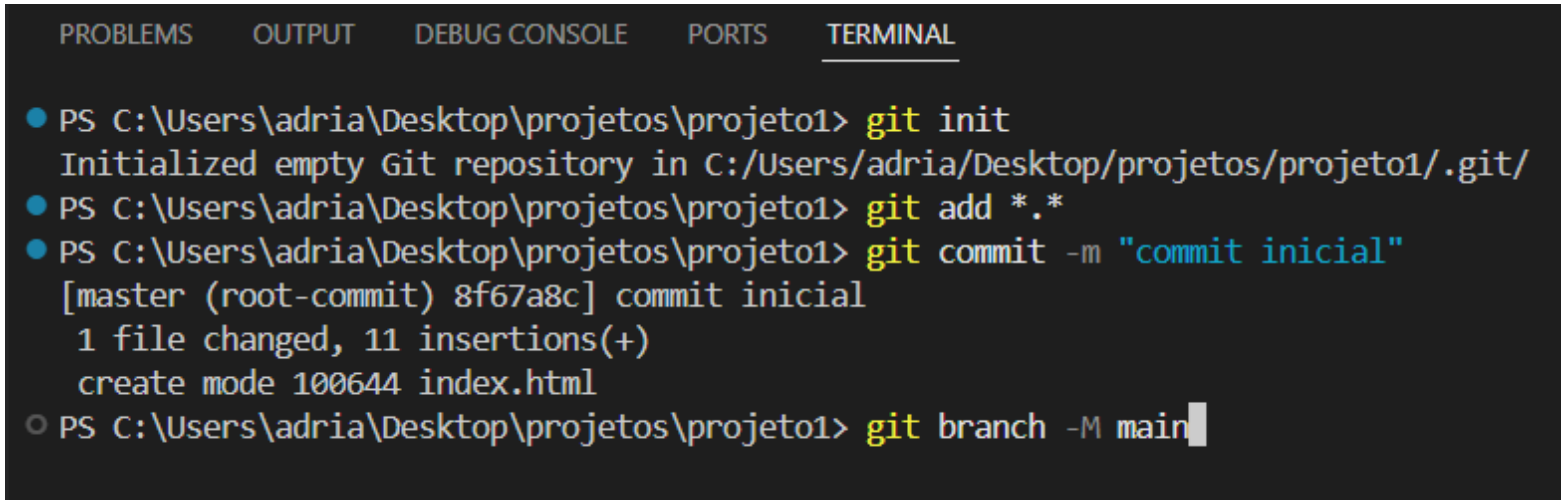
Para especificar o tipo de *commit*, pode usar prefixos no assunto, como:

- **feat** para uma nova *feature*
- **fix** para a resolução de um *bug*
- **style** para recursos e atualizações relacionadas à estilização
- **refactor** para refatoração de uma secção específica da base de código
- **test** para tudo o que está relacionado com testes
- **docs** para tudo o que está relacionado com documentação

Pode usar o comando `git commit` sem uma mensagem ou opção para abrir um editor de texto padrão para escrever a mensagem.

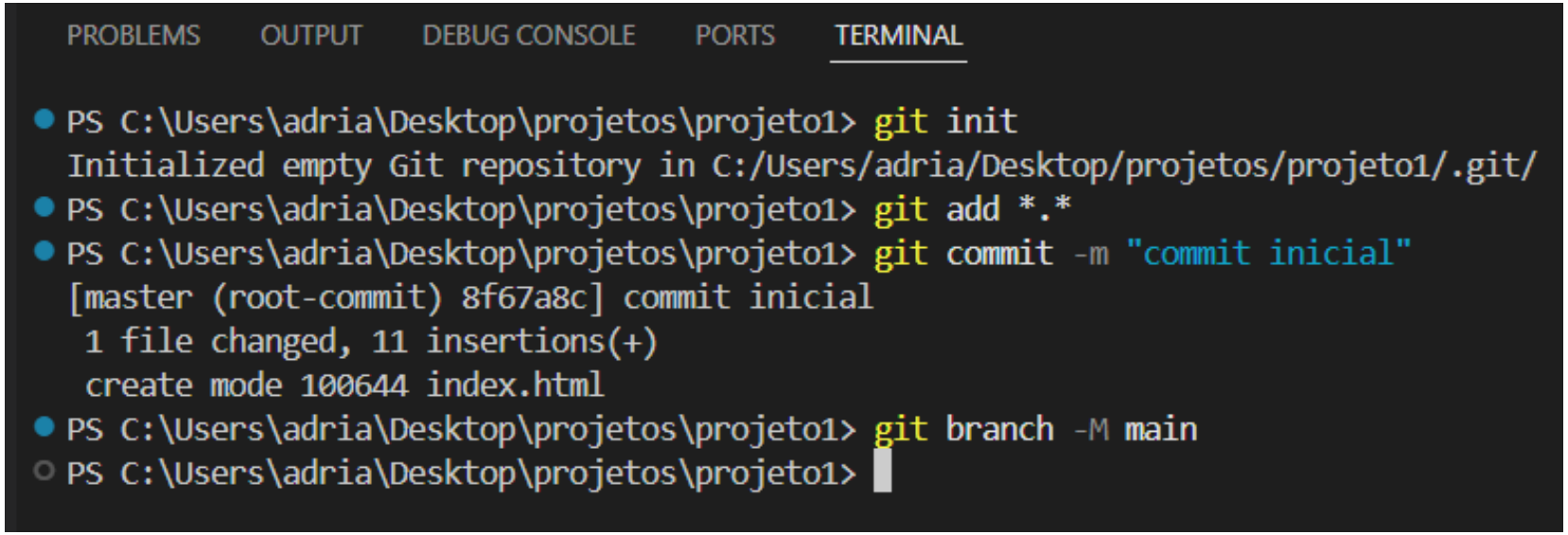
GITHUB – EFETUANDO *BRANCH* REPOSITÓRIOS LOCAIS (TERMINAL)

Agora vamos criar a *branch*, para isso vamos digitar: `git branch –M main` e na sequência pressionamos '`enter`'.



GITHUB – EFETUANDO *BRANCH* REPOSITÓRIOS LOCAIS (TERMINAL)

Como resposta, temos:



GITHUB – EFETUANDO *BRANCH* REPOSITÓRIOS LOCAIS (TERMINAL)

BRANCH

Uma *branch*, é como uma versão paralela do seu projeto.

Ela permite que você trabalhe em algo novo (como uma funcionalidade ou correção de *bug*) sem mudar o código original.

GITHUB – EFETUANDO *BRANCH* REPOSITÓRIOS LOCAIS (TERMINAL)

BRANCH - Exemplo

Imagine que você tem um caderno.

A *branch* principal é como a página onde você anota tudo. Mas, se quiser testar algo novo, você copia a página e escreve nessa cópia. Se funcionar, você cola a cópia de volta na original.

GITHUB – EFETUANDO *BRANCH* REPOSITÓRIOS LOCAIS (TERMINAL)

BRANCH - Exemplo

No Git:

- **Main branch**: É o código principal (o "original").
- **Nova branch**: É uma "cópia" do código para testar ou criar algo novo.

Assim, as *branches* ajudam a organizar o trabalho sem bagunçar o projeto principal.

GITHUB – EFETUANDO *BRANCH* REPOSITÓRIOS LOCAIS (TERMINAL)

No caso do nosso comando `git branch -M main`, faz duas coisas de forma simples:

- Renomeia a *branch* atual para “main” (caso o nome dela seja diferente, como "master").
- Garante que “main” seja a *branch* principal do repositório.

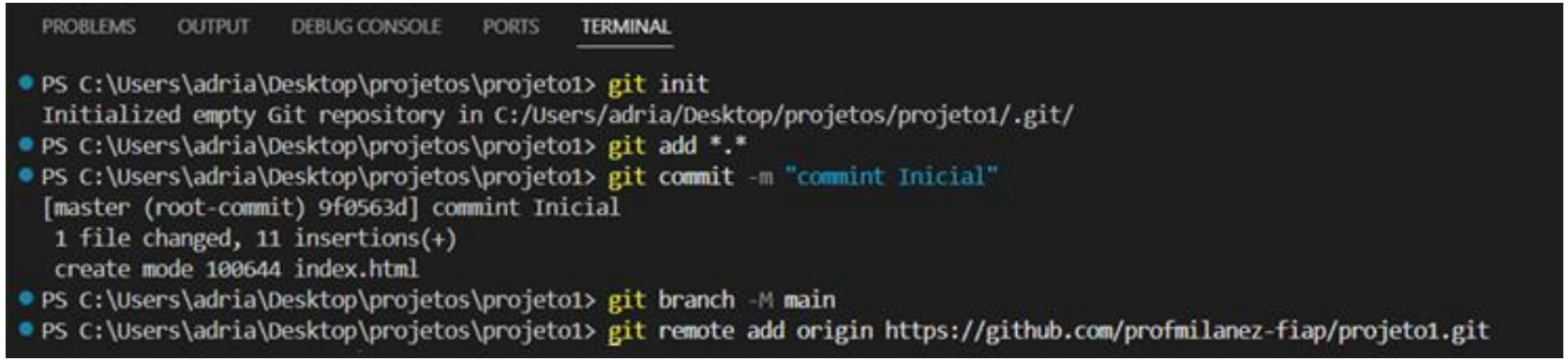
Em resumo:

Esse comando só muda ou confirma que o nome da *branch* principal é “main”.

GITHUB – PUBLICANDO REPOSITÓRIOS LOCAIS (TERMINAL)

Vamos vincular o repositório local ao repositório remoto.

Vamos digitar o comando: `git remote add origin https://github.com/profmilanez-fiap/projeto1.git` e na sequência pressionamos '`enter`'.



GITHUB – PUBLICANDO REPOSITÓRIOS LOCAIS (TERMINAL)

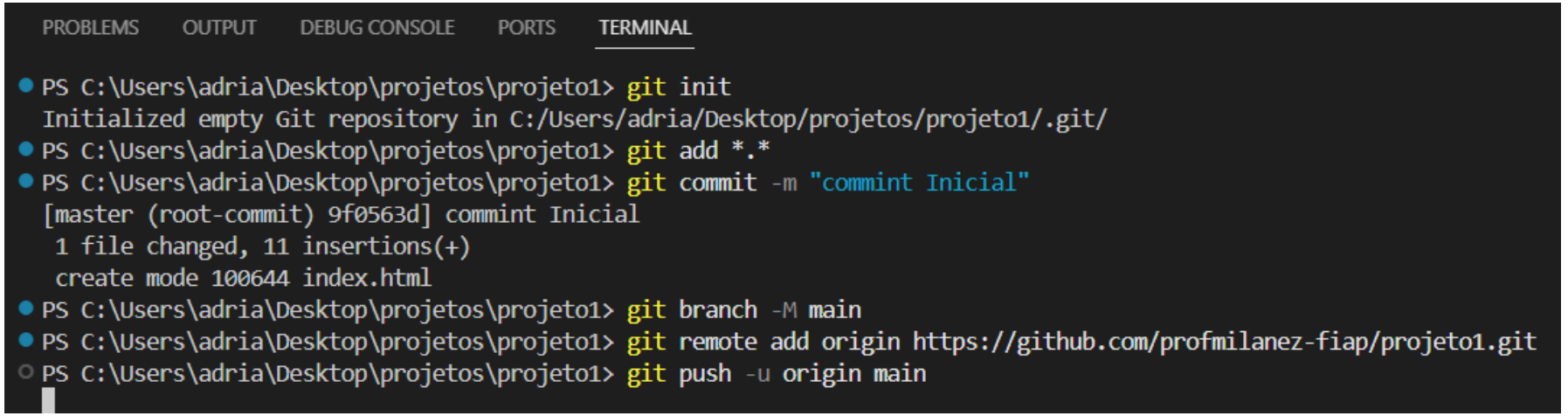
Como resposta, temos:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL

PS C:\Users\adria\Desktop\projetos\projeto1> git init
Initialized empty Git repository in C:/Users/adria/Desktop/projetos/projeto1/.git/
PS C:\Users\adria\Desktop\projetos\projeto1> git add *.*
PS C:\Users\adria\Desktop\projetos\projeto1> git commit -m "commint Inicial"
[master (root-commit) 9f0563d] commint Inicial
 1 file changed, 11 insertions(+)
 create mode 100644 index.html
PS C:\Users\adria\Desktop\projetos\projeto1> git branch -M main
PS C:\Users\adria\Desktop\projetos\projeto1> git remote add origin https://github.com/profmilanez-fiap/projeto1.git
PS C:\Users\adria\Desktop\projetos\projeto1>
```

GITHUB – PUBLICANDO REPOSITÓRIOS LOCAIS (TERMINAL)

E por fim publicamos o(s) arquivo(s) com o comando: `git -u origin main` e na sequência pressionamos '`enter`'.



GITHUB – PUBLICANDO REPOSITÓRIOS LOCAIS (TERMINAL)

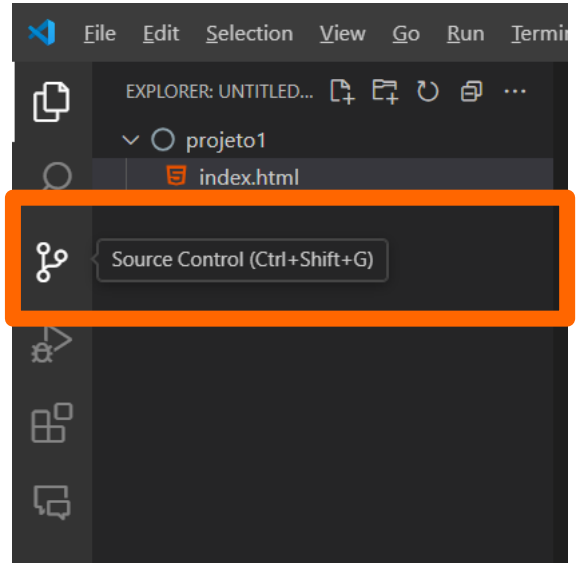
Como resposta, temos:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   PORTS   TERMINAL

PS C:\Users\adria\Desktop\projetos\projeto1> git init
Initialized empty Git repository in C:/Users/adria/Desktop/projetos/projeto1/.git/
PS C:\Users\adria\Desktop\projetos\projeto1> git add *.*
PS C:\Users\adria\Desktop\projetos\projeto1> git commit -m "commit inicial"
[master (root-commit) e1ee940] commit inicial
 1 file changed, 11 insertions(+)
 create mode 100644 index.html
PS C:\Users\adria\Desktop\projetos\projeto1> git branch -M main
PS C:\Users\adria\Desktop\projetos\projeto1> git remote add origin https://github.com/profmilanez-fiap/projeto1.git
PS C:\Users\adria\Desktop\projetos\projeto1> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 368 bytes | 368.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/profmilanez-fiap/projeto1.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\adria\Desktop\projetos\projeto1>
```

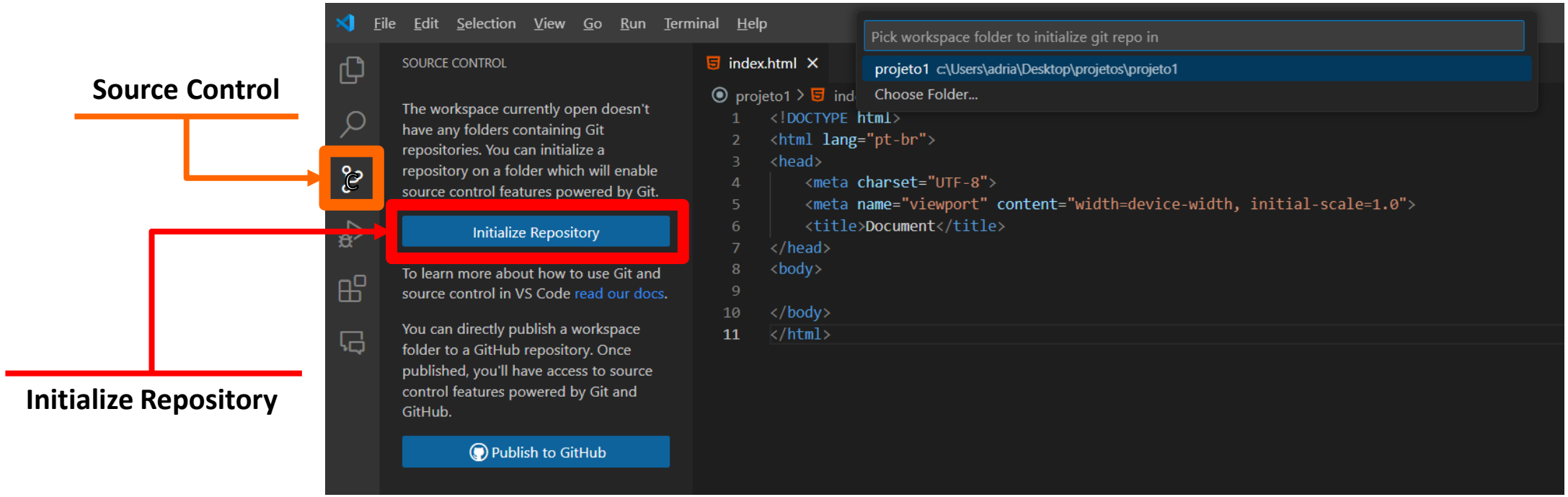
GITHUB – ADD ARQUIVOS EM REPOSITÓRIOS LOCAIS (VSCODE)

Para fazer a mesma coisa, utilizando o recurso '**Source Control**', não precisamos ter a repositório criado antecipadamente no site do **GITHUB**.



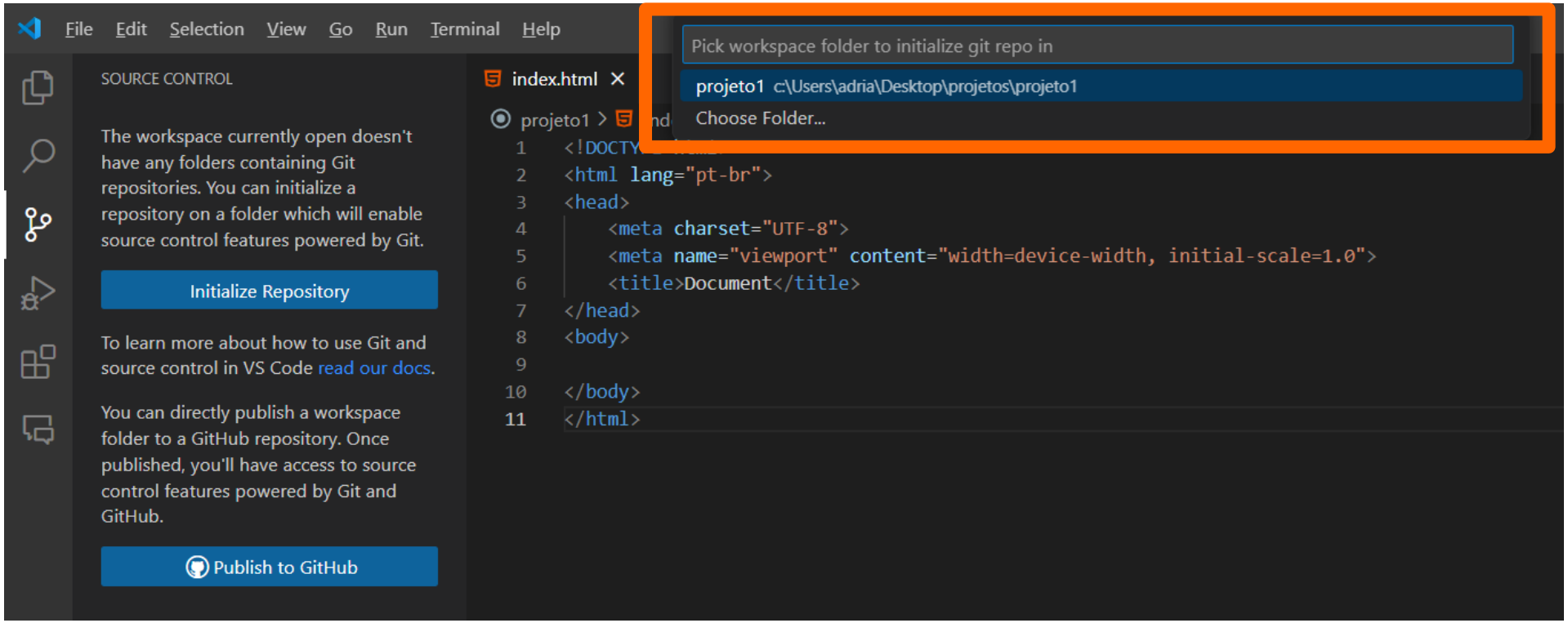
GITHUB – ADD ARQUIVOS EM REPOSITÓRIOS LOCAIS (VSCode)

Precisamos inicializar o repositório localmente. Vamos clicar em 'Source Control' e na sequência, clicamos em 'Initialize Repository'



GITHUB – ADD ARQUIVOS EM REPOSITÓRIOS LOCAIS (VSCode)

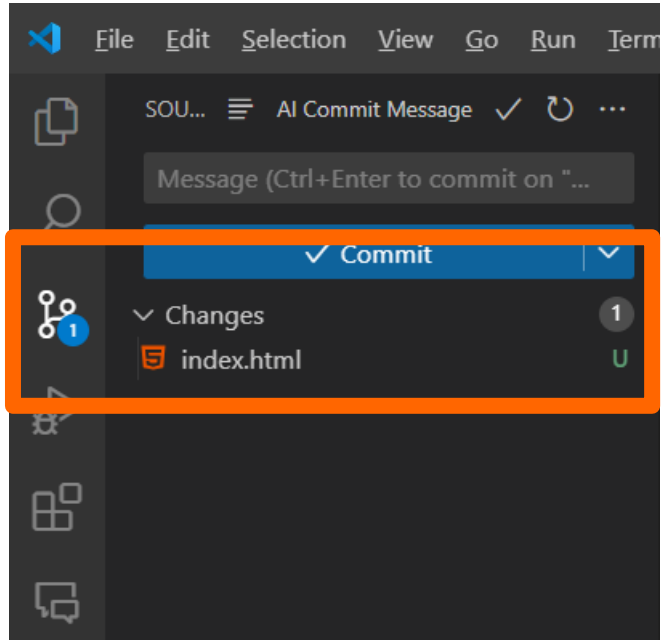
Na sequência, precisamos mostrar se desejamos trabalhar com o repositório que já está aberto ou se desejamos trabalhar com outro repositório.



GITHUB – ADD arquivos em repositórios locais (VSCode)

Após selecionado o repositório local, o(s) arquivo(s) prontos para serem enviados ao repositório remoto deverão ser exibidos como na imagem.

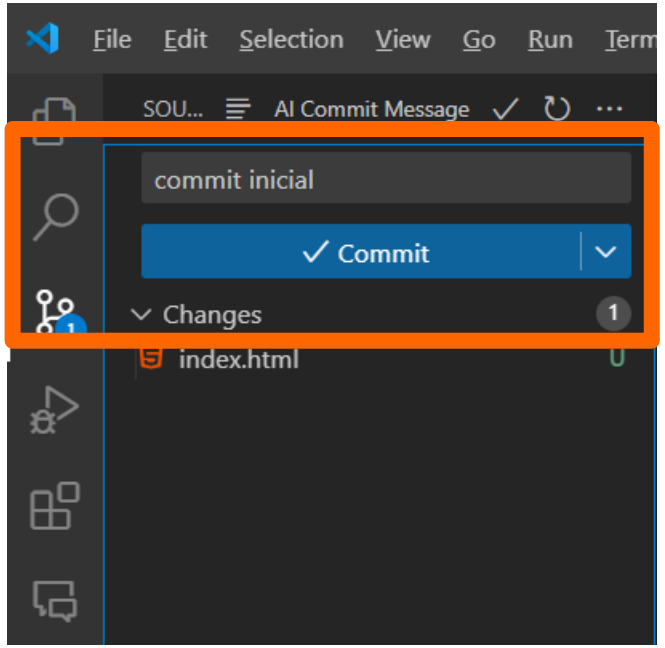
Fazendo uma comparação com o envio do repositório, via Terminal, aqui, fizemos o `git init` e `git add *.* .`



GITHUB – COMMITANDO REPOSITÓRIOS LOCAIS (VSCODE)

Devemos, agora, criar a mensagem para o *commit*. Vamos inserir a mensagem 'commit inicial'.

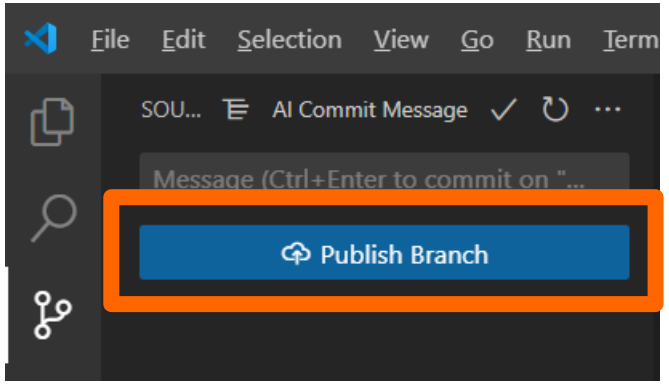
Fazendo a comparação novamente com o envio do repositório, via Terminal, aqui, fizemos o `git commit -m "commit inicial".`



Mais informações sobre *commit*, volte ao slide 39.

GITHUB – EFETUANDO *BRANCH* REPOSITÓRIOS LOCAIS (VSCODE)

Em seguida, devemos clicar 'Publish Branch' para que possamos, de fato, copiar e publicar nosso repositório local no repositório remoto.



Comparando novamente com o envio do repositório, via Terminal, aqui, fizemos o

`git branch -M main` e

`git push -u origin main`

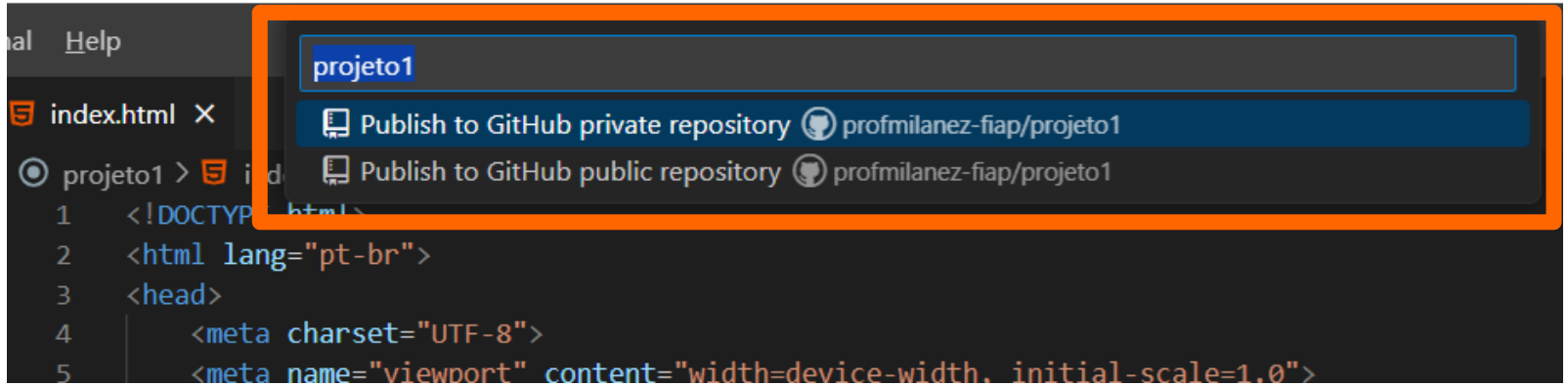
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL

PS C:\Users\adria\Desktop\projetos\projeto1> git init
Initialized empty Git repository in C:/Users/adria/Desktop/projetos/projeto1/.git/
PS C:\Users\adria\Desktop\projetos\projeto1> git add *.*
PS C:\Users\adria\Desktop\projetos\projeto1> git commit -m "commit inicial"
[master (root-commit) 8f67a8c] commit inicial
 1 file changed, 11 insertions(+)
 create mode 100644 index.html
PS C:\Users\adria\Desktop\projetos\projeto1> git branch -M main
PS C:\Users\adria\Desktop\projetos\projeto1> git push -u origin main
```

Mais informações sobre *branch*, volte ao slide 43.

GITHUB – CRIANDO REPOSITÓRIOS LOCAIS

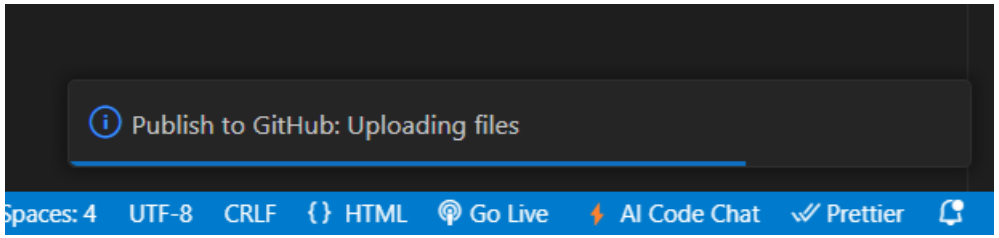
Devemos escolher qual a visibilidade que queremos dar ao novo repositório.



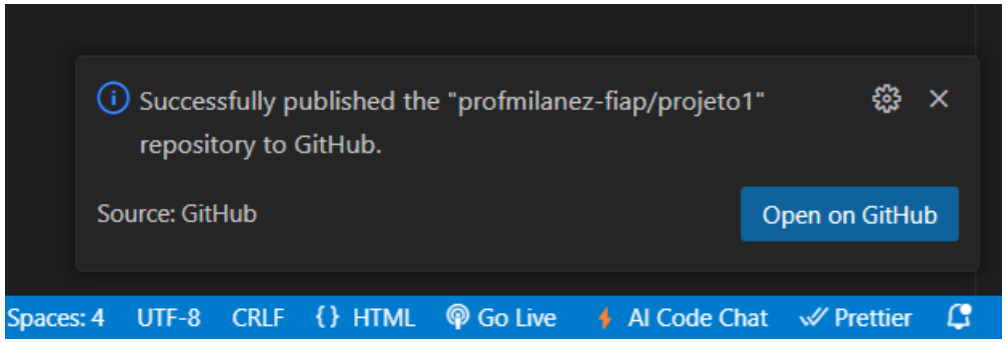
Sobre a visibilidade do repositório remoto, vimos esse assunto no slide 31.

GITHUB – CRIANDO REPOSITÓRIOS LOCAIS

Depois de escolhida a visibilidade, nossos arquivos serão enviados ao repositório remoto.

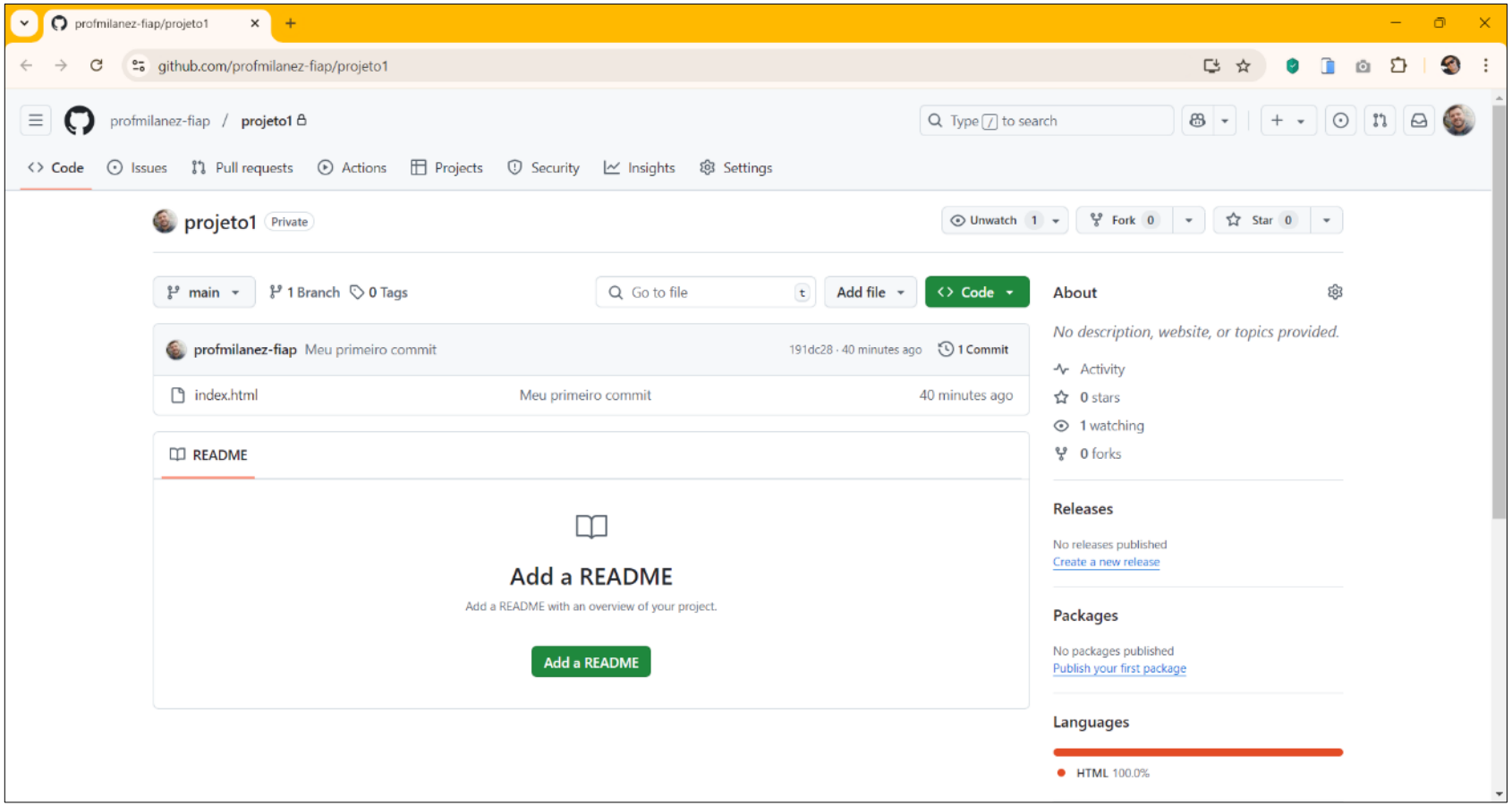


E depois de copiados, temos a mensagem de sucesso.



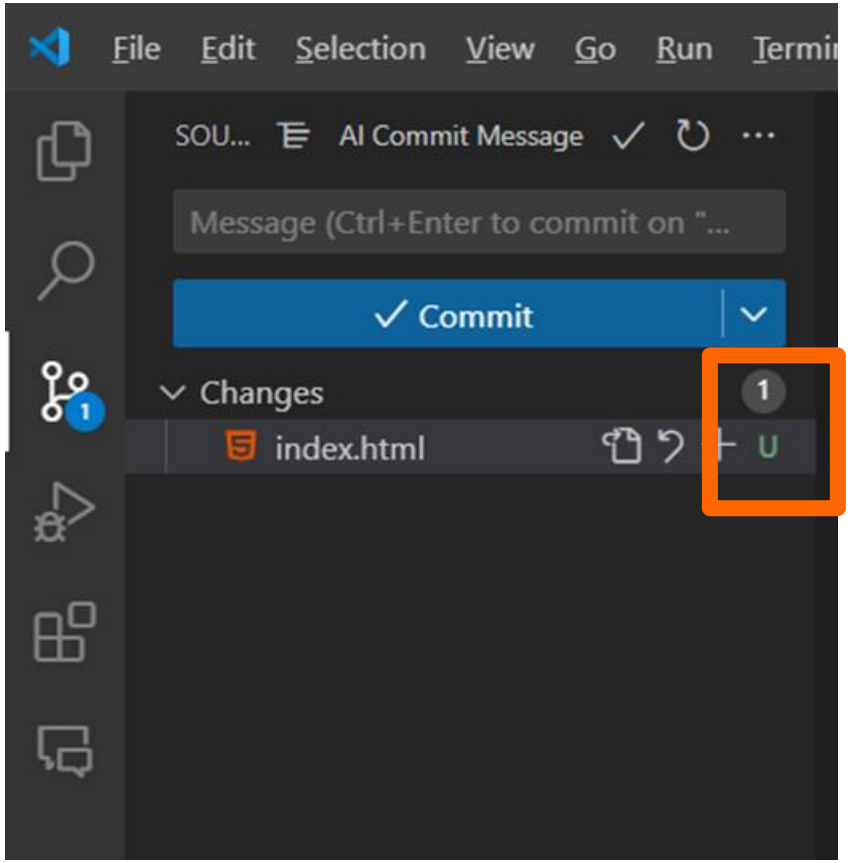
GITHUB – CRIANDO REPOSITÓRIOS LOCAIS

Abrindo o repositório no **GITHUB**, temos.



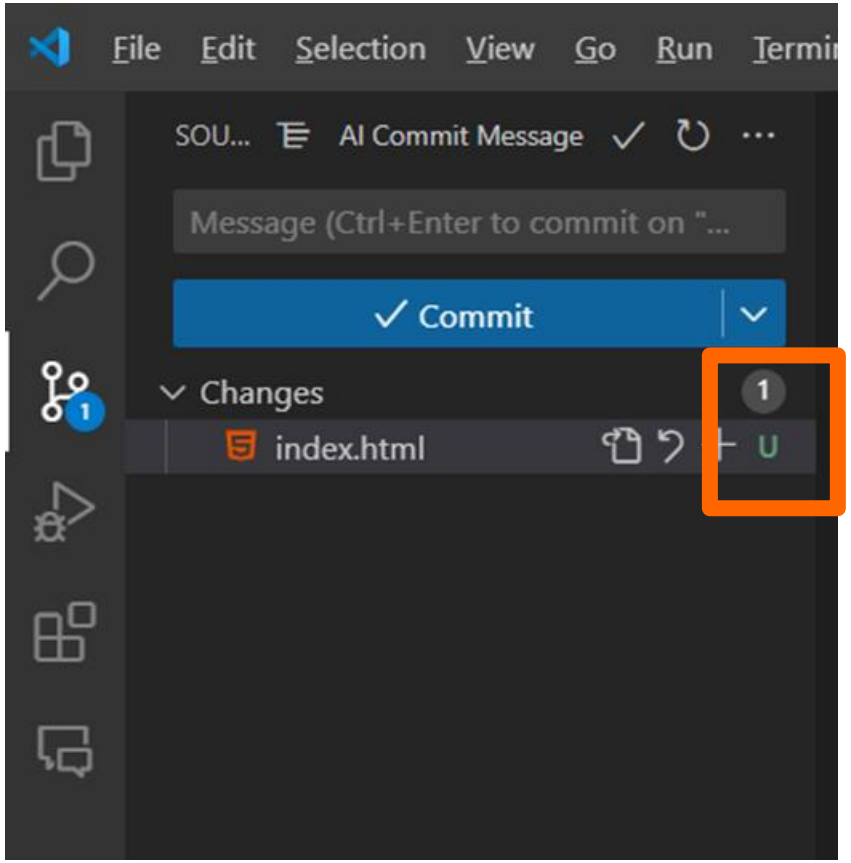
GITHUB – ESTÁGIOS DO GIT

Repararam que ao lado do arquivo existe uma letra ‘U’?

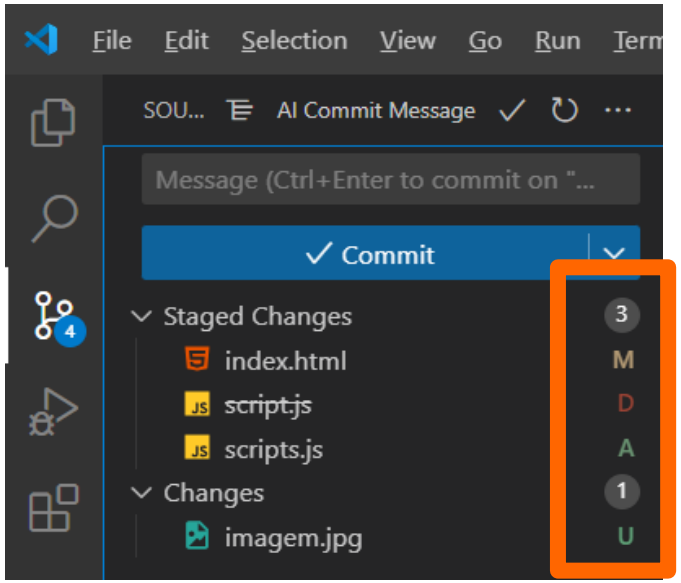


GITHUB – ESTÁGIOS DO GIT

Esse ‘U’ mostra um dos **três** estágios do **GIT**



GITHUB – ESTÁGIOS DO GIT



Estágios do **GIT**

- **U** (**Untracked**): São os arquivos que ainda não são monitorados pelo **GIT**.
- **M** (**Modified**): São os arquivos que já foram *commitados* anteriormente e agora foram modificados.
- **A** (**Added**): São os arquivos que ainda não foram *commitados* mas você já rodou **git add**.
- **D** (**Deleted**): São arquivos que foram excluídos, porém, este não é um estágio do **GIT**. Este na verdade não é estágio do **GIT.**

GIT E GITHUB

SUGESTÃO DE ESTUDO

GIT E GITHUB

Aqui está uma sugestão de tópicos organizados por nível de dificuldade. Isso vai te ajudar a evoluir gradualmente e ganhar confiança com as ferramentas:

1 Conceitos Básicos de GIT

Comece entendendo o que o **GIT** faz e como ele funciona:

- O que é o **GIT** e como ele funciona (sistema de controle de versão).

GIT E GITHUB

- **Instalar e configurar o GIT**
  - git config --global user.name
  - git config --global user.email.

- **Comandos básicos para começar**
  - git init: Inicializar um repositório.
  - git add: Adicionar arquivos ao "estágio".
  - git commit: Salvar alterações no repositório.
  - git status: Verificar o status do repositório.
  - git log: Ver o histórico de commits.

GIT E GITHUB

2 Trabalhando com Repositórios Remotos no GITHUB

Depois de dominar os comandos básicos, conecte o trabalho local ao GitHub:

- **Criar repositórios no GITHUB**
- **Conectar o repositório local ao remoto**
  - git remote add origin URL.
  - git push e git pull.
- **Clonar repositórios existentes**
  - git clone URL.

GIT E GITHUB

3 Branches e Colaboração

Aprenda a trabalhar em equipe ou testar ideias sem interferir no código principal:

- **Criar e alternar branches**
  - git branch NOME_DA_BRANCH e git checkout NOME_DA_BRANCH.
  - git switch (comando mais moderno).
- **Unir branches (merge)**
  - git merge.
- **Resolver conflitos de merge**

GIT E GITHUB

4 Controle Avançado

Explore ferramentas para corrigir erros ou organizar o histórico:

- **Desfazer mudanças**
  - git checkout ou git restore.
  - git reset (diferentes níveis: soft, mixed, hard).
- **Reescrever commits**
  - git commit --amend para corrigir o último commit.
  - git rebase para reorganizar commits.
- **Stash**
  - git stash para salvar mudanças temporariamente.

GIT E GITHUB

5 Fluxo de Trabalho com GITHUB

Aproveite ao máximo os recursos do **GITHUB**:

- **Pull Requests**
  - Como criar e revisar mudanças.
- **Issues e projetos**
  - Organizar tarefas e acompanhar o progresso.
- **GitHub Actions**
  - Automatizar tarefas como testes ou deploys.
- **Markdown**
  - Escrever arquivos README.md atraentes.

GIT E GITHUB

6 Conceitos e Ferramentas Extras

Se já se sente confortável, explore tópicos mais avançados:

- **Hooks do GIT**
  - Automatizar tarefas ao fazer commits/pushes.
- **Cherry-pick**
  - Pegar commits específicos de outras branches.
- **Tags**
  - Marcar versões do projeto.
- **Worktrees**
  - Trabalhar com múltiplas branches no mesmo repositório local.

Referências Bibliográficas

ABBA, Ihechikara. **Tutorial de Git e GitHub – controle de versão para iniciantes**. Tradução: Paula Flávia Pagotto Simionato. FreeCodeCamp, 2022. Disponível em <https://www.freecodecamp.org/portuguese/news/tutorial-de-git-e-github-controle-de-versao-para-iniciantes/>. Acesso em 19 de dezembro de 2024.

BONFIM, Gabriela Pimenta. **Entenda o ciclo de vida dos arquivos no Git e facilite seu trabalho**. 4Linux, 2017. Disponível em <https://blog.4linux.com.br/git-ciclo-de-vida/>. Acesso em 19 de dezembro de 2024.

CHACON, Scott; STRAUB, Ben. **Pro Git**. 2. ed. Nova York: Apress, 2014. Disponível em: <https://git-scm.com/book/en/v2>. Acesso em: 19 dez. 2024.

CARDOSO, Rodrigo. **Git Branch: como usar e gerar ramificações**. Locaweb, 2022. Disponível em <https://www.locaweb.com.br/blog/temas/codigo-aberto/git-branch-como-usar-e-gerar-ramificacoes/>. Acesso em 19 de dezembro de 2024.

GITHUB. **Sobre repositórios**. Github. Disponível em <https://docs.github.com/pt/repositories/creating-and-managing-repositories/about-repositories>. Acesso em 19 de dezembro de 2024.

MELLO, Diego. **Como usar o GitHub? [Guia para Iniciantes]**. Tecnoblog, 2021. Disponível em: <https://tecnoblog.net/responde/como-usar-o-github-guia-para-iniciantes/>. Acesso em 19 de dezembro de 2024.

MICROSOFT. **Introduction to Git in VS Code**. Visual Studio Code, 2024. Disponível em: <https://code.visualstudio.com/docs/sourcecontrol/intro-to-git>. Acesso em 19 de dezembro de 2024.

WILLIANS, Weslley. **Git e GitHub para iniciantes – Tutorial completo**. FullCycle, 2019. Disponível em <https://fullcycle.com.br/git-e-github/>. Acesso em 19 de dezembro de 2024.

DÚVIDAS?

CRÍTICAS?

SUGESTÕES?

AMEAÇAS?